eip ESCUELA INTERNACIONAL POSGRADOS

Máster Avanzado de Programación en Pythn para Hacking, BigData y Machine Learning

Programación Avanzada Python

LECCIÓN 04

Clases y objetos

ÍNDICE

INTRODUCCIÓN

En este cuarta lección vamos a estudiar como trabajar con clases y objetos. Para ello veremos como trabajar con el paradigma POO.

Estudiaremos como trabajar con las clases y todos los elementos que la forman. Y por último, veremos las propiedades que nos ofrece la POO a la hora de programar con este paradigma.

OBJETIVOS

Al finalizar esta lección serás capaz de:

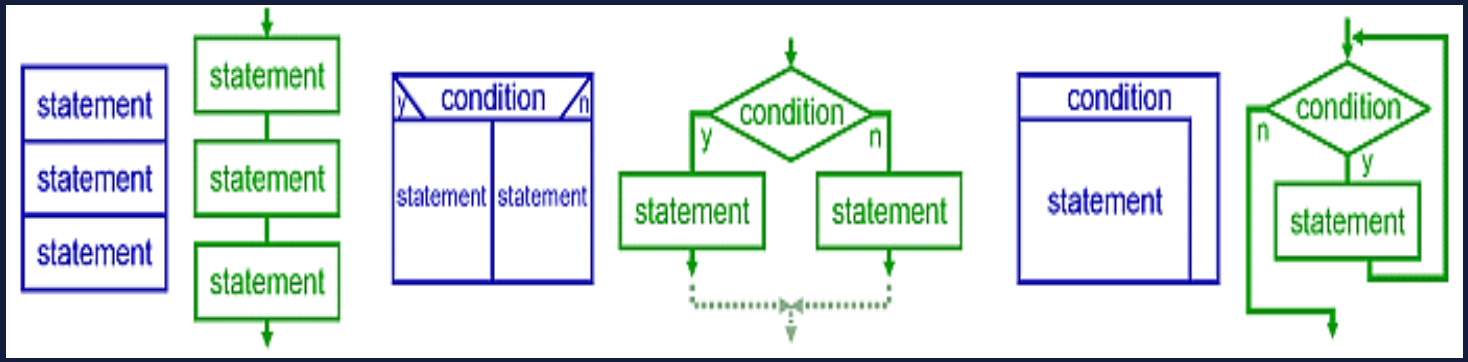
1. Diferenciar entre POO y programación estructurada
2. Trabajar con Clases, objetos y métodos.
3. Fundamentos de POO
4. Aplicar POO en Python

Programación estructurada frente a POO

Un programa puede definirse con un conjunto de estructuras, módulos y funciones. Este tipo de programación es la **programación estructurada**.

En la programación estructurada existen estos 3 tipos de sentencias:

- Sentencias de selección
- Sentencias de iteración
- Sentencias de secuencia



En la programación orientada a objetos, los programas interactúan principalmente con objetos. Los objetos se basan en clases creadas en el programa. Los atributos y métodos también se crean en el programa y se realizan utilizando sus objetos de clase.

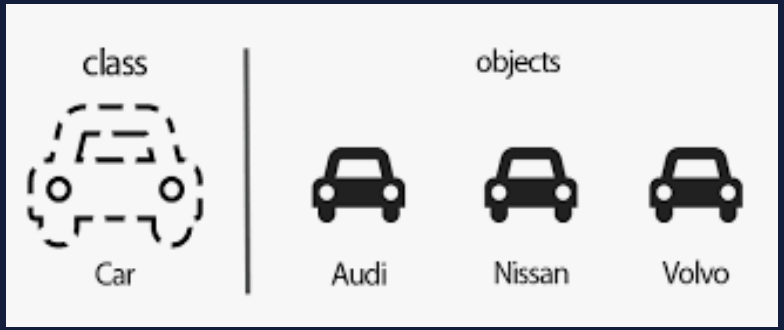
Hay 4 conceptos principales para la programación POO:

- Encapsulación
- Abstracción
- Polimorfismo
- Herencia

Principalmente, las clases contienen:

- Atributos
- Métodos

Las clases trabajan con objetos:



Programación estructurada

- El programa esta dividido en un conjunto de subprograma denominados funciones o módulos
- La modificación de todo el contenido es difícil
- Las funciones interactúan en todo el programa pasando los valores por parámetros y devolviendo las salidas.
- Las funciones son generales, no se utilizan especificadores de acceso.
- Los datos no son seguros
- Código difícil de reutilizar
- Enfoque descendente
- Diseño del programa basado en el proceso
- No se permite la sobrecarga de datos
- Menos flexible
- Menor abstracción
- Lenguajes: C, basic, fortran, pascal

Programación orientada a objetos

- El programa contiene objetos de clases con atributos y métodos
- Es fácil modificar el programa y fácil hacer cambios a través de los objetos de las clases
- La interacción del programa se realiza principalmente con objetos y los valores son pasados por mensajes a través de objetos de clases.
- En las clases se utilizan especificadores de acceso como public, private y protected.
- Datos seguros
- El código es fácil de reutilizar
- Enfoque ascendente
- Diseño del programa basado en los datos
- Se permite la sobrecarga de datos
- Más flexible
- Mayor abstracción
- Lenguajes: Python, php, C#, C++, java, ruby, java script

Clases y objetos

Una clase contiene atributos y métodos. Un objeto es formado mediante la clase, los atributos y métodos.

La clase es creada con la palabra clave 'class'

Sintaxis:

```
class class_name:
        statements
```

Objetos creados como:

```
obj1 = class_name()
obj2 = class_name()
```

Una clase contiene atributos y métodos. Un objeto es formado mediante la clase, los atributos y métodos.

La clase es creada con la palabra clave 'class'

Sintaxis:

```
class class_name:
        statements
```

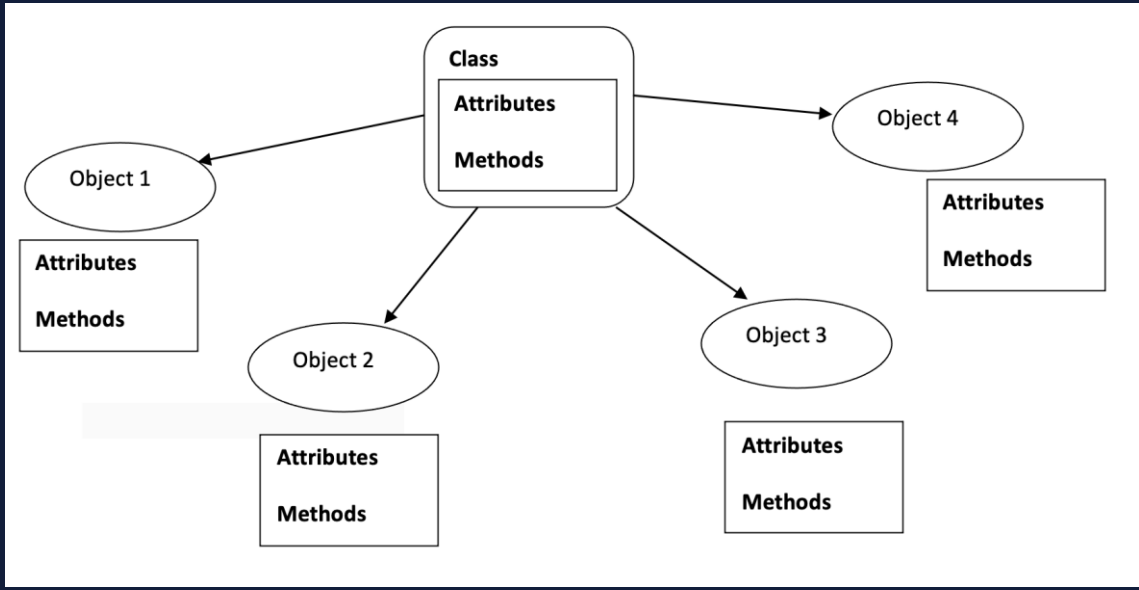
Para crear objetos:

```
obj1 = class_name()
obj2 = class_name()
```

Atributos y métodos

Atributos - tipos de datos variables que se definen e inicializan con valores.

Métodos - funciones que se definen y realizan el calculo con los atributos especificados en la clase



Método init

- El método init es un método especial que comienza y termina con un doble guión bajo como __init__.
- Es el primer método dentro de la clase
- Funciona como constructor de la clase.
- Cuando el objeto de la clase es llamado automáticamente ejecuta la función init.

```
class person:

  def __init__(self,name,age):
    self.name = name
    self.age = age

  def output_display(self):
    print("Name :" , self.name)
    print("Age :", self.age)
```

Variables de clase

Hay dos tipos de variables o atributos que se pueden definir dentro de cada clase, las variables de instancia y las variables de clase

Variables de instancia

Las variables de instancia se definen dentro del método init y son únicas para cada instancia, no son compartidas para todas las instancias u objetos.

Variables de clase

Las variables de clase son compartidas por cada objeto. Se definen en la clase y fuera de cada función.

Variables de clase

```
class Car:

    vechicle = 'car'

    def __init__(self, model, color):
        self.model = model
        self.color = color
```

En la clase car, vehicle = ‘car’ es una variable de clase que se define dentro de la clase y fuera de cada función. Así que es común para cada objeto de la clase.

Las variables model y color son variables de instancia que son únicas para cada objeto cuando se crea el objeto.

Método init

- El método init es un método especial que comienza y termina con un doble guión bajo como __init__.
- Es el primer método dentro de la clase
- Funciona como constructor de la clase.
- Cuando el objeto de la clase es llamado automáticamente ejecuta la función init.

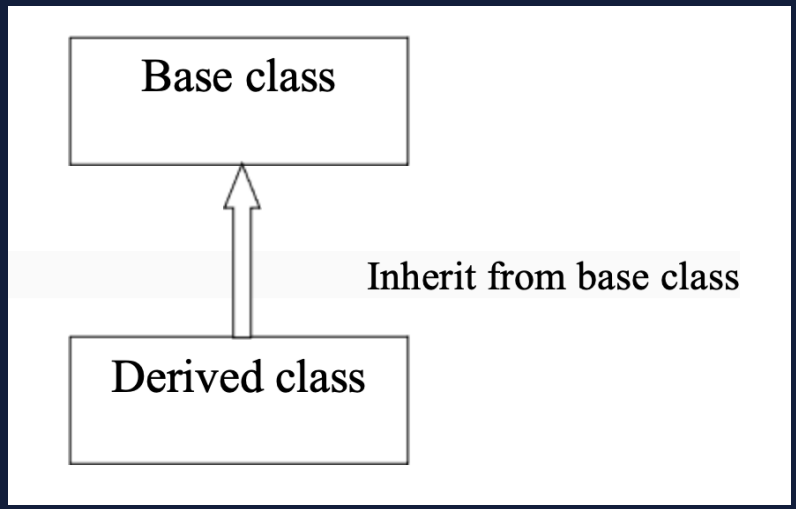
```
class person:

  def __init__(self,name,age):
    self.name = name
    self.age = age

  def output_display(self):
    print("Name :" , self.name)
    print("Age :", self.age)
```

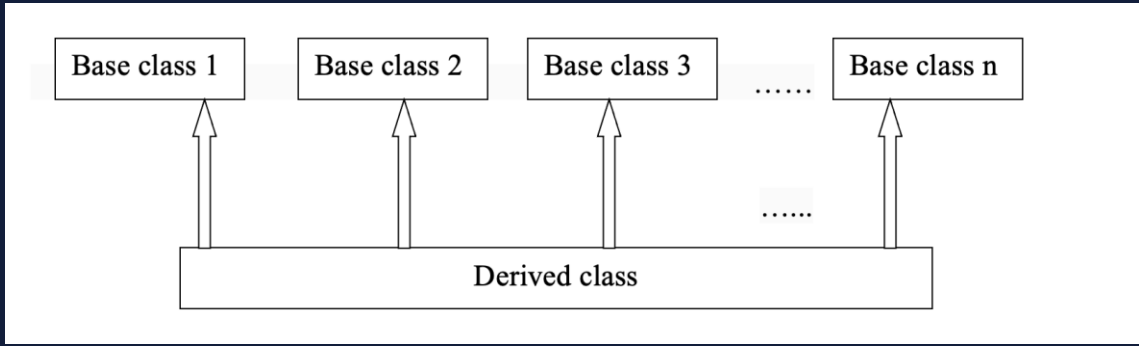
Herencia

Si queremos construir una clase con las características de otra clase y queremos algunas características extras, la clase puede usar las propiedades de la clase original sin reconstruirla de nuevo.



Múltiples herencias

Una clase derivada puede heredar las propiedades de varias clases base. La clase derivada podrá acceder a los atributos y métodos de las clases base.



Múltiples herencias

```
class addition():
  def add(self,x,y):
    return x+y

class subtraction():
  def sub(self,x,y):
    return x-y

class calcualte(addition,subtraction):
  def mul(self,x,y):
    return x*y
```

```
result = calcualte()

print(result.add(3,2))
print(result.sub(3,2))
print(result.mul(3,2))
```

```
5
1
6
```

Sobreescritura de métodos

Normalmente heredamos los métodos de la clase base a la clase derivada. En algunos casos, debe haber algunas modificaciones o cambios necesarios para los métodos de la clase base. La clase derivada puede definir de nuevo la misma clase con modificaciones

```
class speaking():
  def speak(self):
    print("I am speaking")

class sleeping(speaking):
  def speak(self):
    print("I am sleeping")
```

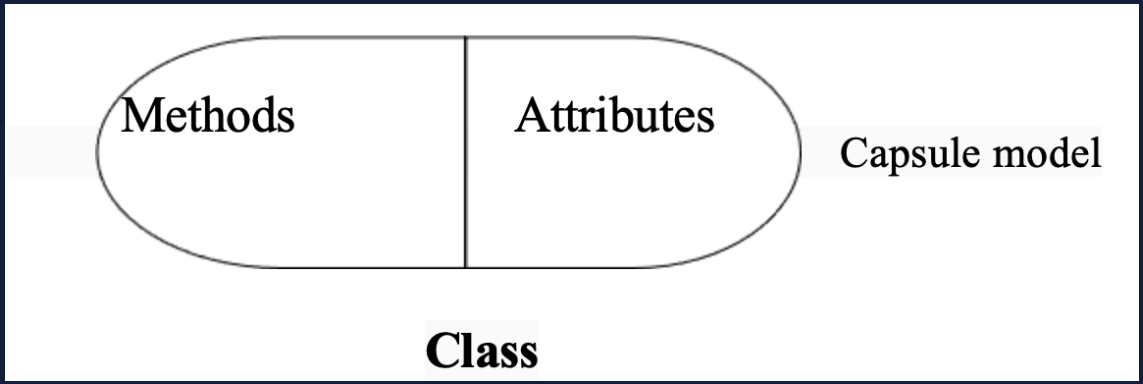
```
object = sleeping()

object.speak()
```

```
I am sleeping
```

Encapsulación

Mediante la encapsulación se consigue que los atributos y métodos oculten los datos al exterior.



Encapsulación

Hay 3 tipos de modificadores de acceso:

- Public
- Private
- Protec

En el encapsulamiento, sólo se utilizan los modificadores *private* y *protect* para que los datos sólo sean accesibles por su clase

Miembros protegidos

Los miembros protegidos comienzan con un guion bajo, '_'

```
class base_class:

    def __init__(self):
        self._protect = 2
```

La variable protegida es accedida por su propia clase derivada. Si llamamos directamente a la variable *protect* fuera de la clase nos dará un error.

Miembros privados

Los miembros privados comienzan con un doble guion bajo, '__'

```
class base_class:

    def __init__(self):
        self.__protect = 2
```

La variable protegida se define como privada, cuando una clase derivada llama al miembro privado nos dará un error.

Polimorfismo

Polimorfismo significa diferentes formas en distintas condiciones. En el entorno de Python significa que las propiedades como métodos y entidades muestran diferentes formas o resultados en diferentes tipos.

```
a = 100
b = 3.4333
sum = a + b
print(sum)
```

```
103.4333
```

```
str1 = "hello "
str2 = "xyz"
str3 = str1 + str2

print("String after concatenation:", str3)
```

```
print(len("string"))
print(len(['list_item1','list_item2']))
print(len({'key1':'value1','key2':'value2'}))
```

```
6
2
2
```

Abstracción

La abstracción es otro de los conceptos de la POO y se utiliza para ocultar los datos del exterior. Los datos se abstraen usando el modificador de acceso private.

Los atributos que comienzan con doble guion bajo '__' no pueden ser accedidos desde el exterior de la clase.

CONCLUSIONES

1. La **programación estructurada** Un programa puede definirse con un conjunto de estructuras, módulos y funciones.

2. En la **programación orientada a objetos,** los programas interactúan principalmente con objetos. Los objetos se basan en clases creadas en el programa.

3. Una **clase** contiene **atributos** y **métodos**. Un objeto es formado mediante la clase, los atributos y métodos.

MUCHAS GRACIAS POR SU ATENCIÓN

eip ESCUELA INTERNACIONAL POSGRADOS

rsanchezi@grupomainjobs.com

Rubén Sánchez Iruela
linkedin.com/in/ruben-sanchez-iruela-8156799a

twitter.com/eiposgrados

facebook.com/eiposgrados

instagram.com/eiposgrados